

4-11-00

A

THE ASSISTANT COMMISSIONER OF PATENTS
Washington, D.C. 20231

DOCKET NUMBER: RAL9-00-0029
APRIL 10, 2000

Sir:

Transmitted herewith for filing is the Patent Application of:

Inventor: CLAUDE BASSO ET AL

For: NETWORK PROCESSOR SERVICES ARCHITECTURE THAT IS PLATFORM AND OPERATING SYSTEM INDEPENDENT

Enclosed are:

☒ Patent Specification and Declaration

☒ 4 sheets of drawing(s).

☐ An assignment of the invention to International Business Machines Corporation (includes Recordation Form Cover Sheet).

☐ A certified copy of a application.

☐ Information Disclosure Statement, PTO 1449 and copies of references.

The filing fee has been calculated as shown below:

For	Number Filed	Number Extra	Rate	Fee
Basic Fee				\$690
Total Claims	20	-20	x 18 =	\$
Indep. Claims	5	- 3	2 x 78 =	\$156
MULTIPLE DEPENDENT CLAIM PRESENTED			x260=	\$
			TOTAL	\$846.00

☒ Please charge IBM Corporation Deposit Account No. 09-0464 in the amount of \$846.00. A duplicate copy of this sheet is enclosed.

☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to IBM Corporation Deposit Account 09-0464. A duplicate copy of this sheet is enclosed.

☒ Any additional filing fees required under 37 CFR §1.16.

☒ Any patent application processing fees under 37 CFR §1.17.

CERTIFICATE OF MAILING BY "EXPRESS MAIL" UNDER 37 CFR § 1.10

"Express Mail" mailing label number EL453464531US:

Date of Mailing: APRIL 10, 2000

I hereby certify that the documents indicated below are being deposited with the United States Postal Service under 37 CFR 1.10 on the date indicated above and are addressed to Box Patent Applications, Assistant Commissioner of Patents, Washington, D.C. 20231 and mailed on the above Date of Mailing with the above "Express Mail" mailing label number.

Chris Mark

Respectfully submitted,

By EUSTACE P. ISIDORE

Registration No. SEE ATTACHED

FELSMAN, BRADLEY, VADEN, GUNTER & DILLON, LLP

Suite 350 Lakewood on the Park
7600B North Capital of Texas Highway
Austin, Texas 78731
Telephone (512) 343-6116

04/10/00
JCS53 U.S. PTO

JCS564 U.S. PTO
09/546133
04/10/00

NETWORK PROCESSOR SERVICES ARCHITECTURE THAT IS PLATFORM AND OPERATING SYSTEM INDEPENDENT

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates in general to improved computer networks, and in particular to an improved network processor architecture. Still more particularly, the present invention relates to an improved distributed network processor architecture that supports individualized network processor services that are platform and operating system independent.

2. Background of the Invention:

Figure 1A shows a block diagram of a general communications network. The communications network includes switch (or router) **10** connected through a simple multiplex device (multiplexer) **11** to several workstations **13**. Switch **10** is also connected to a control point **12**, Internet **14** or some other form of Wide Area Network (WAN) **16**, server **18**, and a Local Area Network (LAN) **20**. LAN **20** can be configured with an ethernet, token-ring or any other type of architecture, which might include many additional workstations. Those skilled in the art will understand that a great variety of combinations of such elements is possible in permutations of **Figure 1**. In particular, some elements could be duplicated and others

deleted. Multiplexer 11 is a conventional off-the-shelf device that selects one of the plurality of attached workstations and allows it to communicate with the switch. Control point 12 provides the interface between the switch 10 and a network administrator. Control point 12 may include a computer comprising a keyboard, control unit, display, etc.

Communications networks are continually evolving due to the growing demand for networks with larger bandwidth, lower latency and greater flexibility. Many limitations exist in traditional networks. For example, networks designed with custom design chips, which carry out specific network applications, are not easily adaptable and are unable to keep up with the constantly evolving requirements for improved technology. Network changes occurs constantly, while fabrication of each custom design chip typically takes several months. Also, networks implemented with general purpose processors are limited in performance by the speed of the processor.

Distributed network systems created with a network processor services architecture are quickly becoming the standard for routing information throughout both Intranet and Internet networks. Figure 1B illustrates a typical network processor services architecture. Control processor 101, which may be synonymous with control point 12 of Figure 1A, directs most of the processes of the system. Control processor 101 is coupled to a plurality of network processors 103 presented as distributed

parallel network processors 103. Network processors 103 are specialized processor chips designed to process networking applications in real time. These processor chips are initialized and configured by the general purpose processors or control processor 101.

Network processor 103 receives frames, modifies them, and provides media interface for transmitting the frames out to switch fabric 105 or to control processor 101. Network processors 103 are designed to optimize network-specific tasks and provide exceptional bandwidth, throughput and distributed processing capabilities, while reducing latency. Network processors 103 receive and forward packets during network communication. The packets are routed via the network processor 103 on layers 2,3, and 4 of network protocols. Network processors 103 and switch fabric 105 may exist as components of switch/router 10 illustrated in Figure 1A. Network processors 103 read the addresses of the packets, translate the addresses based on protocol and route the packets to their correct destination. Switch fabric 105 allows interconnection of multiple network processors in a system

Control processor 101 contains a high level intelligent protocol for controlling the flow of IP packets in the system. Control processor 101 is responsible for activating network processors 103. Control processor 101 also provides network processors

103 with address information, that identifies the destination of the frames. Network processor 103 has one or more multiple look-up tables, which it consults when a packet arrives to determine a forwarding destination for the packet.

While the use of a network processor architecture offers a flexible solution that helps to maximize bandwidth utilization and traffic flow, use of the architecture can be limited to specific processor types and operating systems. Presently, communications within the network processors architecture is processor and OS dependent. The network processor and control processor are tightly coupled with specific device drivers which operate as a unified functional block that is designed to interact with a particular processor hardware type and operating system.

The present invention recognizes that more flexible processor model of network processor products is desired. A network processor services architecture, which was capable of being implemented in a hardware and operating system independent manner would be a welcomed improvement. It would be further desirable if such a processor architecture was expandable to include other functional components, which may be customer specific or developer specific. These and other benefits are achieved by the present invention.

SUMMARY OF THE INVENTION

5 A system for providing a scalable processor and
operating system independent network processor services
architecture is disclosed. The system includes a
plurality of portable and individualized functional
components representing particular segments of the
control processor's device driver. The functional
components, which include lower level and external APIs,
10 carry out the various network processor functions, such
as the receipt and transfer of packets on the network and
other functions required by the control processor to
communicate with and direct the network processor. The
functional components are designed to be adaptable to the
various types of processor architecture and operating
systems available and to permit customers or developers
to customize and expand the available network services.

20 All objects, features, and advantages of the present
invention will become apparent in the following detailed
written description.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention itself, as well as a preferred mode of use, further objects, and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1A depicts a conventional communications network in which a network processor architecture may operate;

Figure 1B illustrates a block diagram of basic components of a network processor architecture;

Figure 2 depicts a block diagram of a computer system in which a preferred embodiment of the present invention may be implemented; and

Figure 3 depicts a high level block diagram representation of the functional components of a device driver within a control processor of a network processor services architecture in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENT

5 The present invention may be executed in a variety
of systems, including a variety of computers running
under a number of different operating systems. The
computer may be, for example, a personal computer, a
network computer, a mid-range computer or a mainframe
computer. In the preferred embodiment, the computer is
10 utilized as a control point of a network processor
services architecture within a local-area network (LAN)
or a wide-area network (WAN).

Referring now to the drawings and in particular to
15 **Figure 2**, there is depicted a block diagram of a typical
computer system 210 that may be utilized as a control
processor in a preferred embodiment of the present
invention. As shown, processor (CPU) 212, Read-Only
memory (ROM) 213, and Random-Access Memory (RAM) 214 are
connected to system bus 211 of computer system 210. CPU
20 212, ROM 213, and RAM 214 are also coupled to Peripheral
Component Interconnect (PCI) local bus 220 of computer
system 210 through PCI host bridge 216. PCI Host Bridge
216 provides a low latency path through which processor
25 212 may directly access PCI devices mapped anywhere
within bus memory and/or input/output (I/O) address
spaces. PCI Host Bridge 216 also provides a high
bandwidth path for allowing PCI devices to directly
access RAM 214.

Also attached to PCI local bus 220 are communications adapter 215, small computer system interface (SCSI) 218, and expansion bus bridge 229. Communications adapter 215 is utilized for connecting computer system 210 to a network 217. SCSI 218 is utilized to control high-speed SCSI disk drive 219. Expansion bus bridge 229, such as a PCI-to-ISA bus bridge, may be utilized for coupling ISA bus 225 to PCI local bus 220. In addition, audio adapter 223 is attached to PCI local bus 220 for controlling audio output through speaker 224. In alternate embodiments of the present invention, additional peripheral components may be added.

Computer system 210 also preferably includes an interface such as a graphical user interface (GUI) and an operating system (OS) that reside within machine readable media to direct the operation of computer system 210. In the preferred embodiment, OS (and GUI) contains additional functional components which permit network processing components to be independent of the OS and/or platform. Any suitable machine-readable media may retain the GUI and OS, such as RAM 214, ROM 213, SCSI disk drive 219, and other disk and/or tape drive (e.g., magnetic diskette, magnetic tape, CD-ROM, optical disk, or other suitable storage media). Any suitable GUI and OS may direct CPU 212.

Further, computer system 210 preferably includes at least one network processor services architecture software utility (i.e., program product) that resides within machine readable media, for example a custom defined service utility 208 within RAM 214. The software utility contains instructions (or code) that when executed on CPU 212 interact with modified OS of the invention to execute at least one block of the functional utilities depicted in Figure 3.

The invention provides a distributed network processor services architecture in which the control functions of the device driver within the control processor are created as portable, exchangeable and individual functional blocks or utilities. The functional blocks operate to provide a network processor services architecture that exhibits operating system and hardware independence. The invention thus provides a device driver functionality which is adaptable to (or usable in) any control processor or operating system environment to control the operation of the network processor.

More specifically, the invention provides a device driver program for a distributed processing system. The device driver program includes a plurality of individual system services programs (or utilities), which provide high level bi-directional communication between a software stack and each network processor. The invention is implemented conceptually as software, which is encoded on a chip or other computer readable medium and connected

to the processor of a network control point (or control processor).

Referring now to **Figure 3**, a preferred implementation of the various processor services as individual functional component blocks is illustrated. As depicted the new configuration provides the separation (or isolation) of all system services of the control processor from the operating system, the separation (or isolation) of the physical transport services in order that the network processor code may reside on the CP, and scalability in that the system processor services reside between the external and low level Application Programming Interface (APIs) and may be replaced or supplemented with new ones. The functional components are illustrated as blocks in **Figure 3** connected by interaction lines. In addition to the interaction lines shown, each of the scalable services can interact with one another. The blocks illustrated and their functionality are described below.

The network processor device driver **301** of **Figure 3** is illustrated with three levels of components, external APIs **303**, low level APIs **323**, and physical transport interface **339**. Within each level are the functional blocks which represent the network processor services. Device driver **305** is the software component that attaches the NP hardware **343** to the software stack **300**. Device driver **305** is responsible for being the interface between network processor **343** and the software stack **300**. Device

driver 305 initializes data structures required for interfacing with each of the interfaces. Thus, in the preferred embodiment, device driver 305 must also know the location of each network processor and know which ports are on which network processor.

DD 305 can be broken down into the upper layer API, Internal device drivers, and the lower level primitives. The upper layer APIs are similar to general APIs of the processor; however the internal structures and lower layer primitives are unique components. Lower layer primitives provides both a control path and a data path. Device Driver (DD) 305 provides the following functions:

1. Initialization of device structure for use by the software stack 300;
2. CP or carrier interface functions;
3. Frame receive and/or transmit functions; and
4. Control message transmit and/or receive.

The physical transport services (PTS) 341 transports the frames to the network processor 343. PTS 341 provides the base Send/Receive API to transmit data and control frames to and from Network Processor 343. This component isolates the higher layer services from the physical connection of the network processor. For example, PTS 341 for an embedded PowerPC within the network processor 343 would be different from and external PowerPC connected to the network processor via an ethernet cable. PTS 341 is coupled to network

processor 343, network processor transport services 325, and operating system (OS) 303.

5 Network Processor 343 is connected to network processor device driver 301 of control processor. Each network processor 103 is a programmable, communications integrated circuit capable of performing one or more of the following functions: (1) packet classification (identifying packets based on known characteristics such as address or protocol); (2) packet modification (modifying the packet to comply with IP, ATM, or other protocols); (3) queue or policy management (packet queuing, de-queuing, and scheduling of packets for specific applications.); and (4) packet forwarding (transmission and receipt of data over the switch fabric and forwarding and routing the packet to appropriate address.). Within network processor 343, pico-code runs on Guided Cell handlers (GCH) 375, Guided Tree handlers (GTH) 377, and General Data handlers (GDH) 379. These software components are responsible for initializing the system, maintaining the forwarding paths, and managing the system.

25 Network processor 343 operates independent of and unaware of the architectural make-up of the network processor device driver 301 (i.e., the independent or individualized component blocks). All frames and control messages, etc. are translated (or decoded) within device driver 301 prior to being transmitted to the network

processor 343 by the physical transport service 341. The translation enables seamless communication between the specific network processor, the operating system 303 and the internal components blocks. Likewise, any frames received from the network processor 343 are transcoded for use by the general service utility blocks of the network processor device driver 301

During operation, the control processor communicates with each NP 343 using a control messaging protocol 269. Software architecture of NP low level and external APIs 323 and 301 supports bridging, routing, ATM, and multi-layer switching. The initialization phase of network processor 343 sets up the registers and memory regions of the processor chip and loads the code for GCH 375. GCH 375 prepares the chip to begin handling frames. To implement this, pico-code, which contains the code instructions used to forward each incoming frame, is loaded onto the chip.

Network Processor Transport Services 325 is located above Physical Transport Services 341 and handles the specific Network Processor control and data encapsulation processing. Network processor transport services 341 further includes the following component blocks:

- (1) Raw 333, which is an interface for connecting to the physical transport service 341

(2) Mux/Demux 331, a processing box (multiplexer) that encapsulates or de-encapsulates the Control and Data frames being sent to and from the NP 343.

(3) Data 327, which represents the generic data frames that may be modified to fit the network processor type; and

(4) Control (or control message services) 329, which is the generic, internal control messages modified and utilized to maintain the state of the Network Processor 343.

Control message services 329 provides basic Network Processor initialization, configuration, code loading, event notification, debugger, diagnostic, and software action services for example. From the control services 329, one or more Network Processor 343 can be controlled in a given system.

The control message formatting services 337 is located in the lower level APIs 323 and is coupled to the external APIs 305, OS 323, software stack 300, and network processor resource services 335. Control message formatting services 337 is a messaging protocol utilized to exchange control messages between the control processor and the network processor 343. Control message formatting services 337 operate as an interpreter within the device driver 301. Control message formatting services 337 converts all low and high level APIs of various components into a specific language utilized by

network processor 343. In the preferred embodiment, control message formatting services 337 is also responsible for converting an OS or hardware specific request into a general request applicable to any OS or hardware and vice versa, which is understood by the other components.

The primary portable functional utilities of the network processor device driver 301 exist in external APIs 305. These functional utilities may exist as firmware or software loaded in the RAM of the computer system or accessible through a computer readable medium. External APIs 305 provide access to and control over the underlying features of Network Processor 343. Network processor services APIs represent the services provided in each network processor subsystem and can be used as a common base to control one or more additional subsystems. The functional blocks or utility of external APIs 305 and their respective functions are provided below.

Custom Defined Services 307 represents a new component addable by a customer or developer based on his specific networking needs. It is conceivable that this element will greatly enhance the scalability of the processor services architecture implemented according to the present invention. Thus, other utilities may be included to expand or replace the external APIs 305 presented in Figure 3 based on network design, customer needs and/or developer's preference. The APIs presented

and described herein are done so solely for illustrative purposes and not meant to be limiting on the invention.

Interface Manager Services 309 may be utilized by developers to create and configure media ports (e.g., Ethernet, POS, and ATM) that will interface with protocol stacks. The ports are connectors on the system that manage, send and receive data within the network. The data is then processed by the network processor 343. In a preferred embodiment, interface manager services 309 also offers two additional features for integrating with the network processors 343. The first feature allows for creation of logical interfaces for receiving and transmitting frames that originate from and are destined for the system itself. Examples of these interfaces are Telnet, file transfer protocol (FTP), and simple network management protocol (SNMP) sessions. The second feature provides a high-speed, highly reliable, and high-bandwidth communications pipe for interprocess communications (IPC) among general purpose processors for systems that will make use of in-band forwarding services of the Network Processor based system.

Table Services 313 are a database of forwarding and/or control information on Network Processor 343. Network Processor 343 fundamentally supports full match, longest prefix match, and multifield classification trees. Table Services APIs 313 provide an application independent way to manage each of the table types. Table management consists of creation, initialization, purging

and basic insertion, deletion, update, and query operations. Table services 313 can be utilized to develop applications that distribute control and topology information to one or more Network Processors 343.

5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995

Network Processor 343 contains a large number of queues that are used for basic chip operation and advanced scheduling functions. Queue Services APIs 317 allow access to these queues for setting and monitoring thresholds, guaranteeing minimum and maximum bandwidth, and creating and initialization of dynamic queues. Queue Services 317 may vary based on the version of network software running on Network Processor 343 and address issues such as Differentiated Services, ATM, and flow control. Like the interface manager services 309, queue services APIs 317 are designed to interface with an application stack 300.

20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995

Protocol services APIs 319 provide protocol stack interfaces for well-known protocols such as Bridging, IP, and MPLS, which Network Processors 343 are utilized to deploy. An example would be the IP Protocol APIs including IP Forwarding Table and ARP Table management. The protocol services APIs 319 provides APIs to control the behavior of the protocol forwarding code. For example, an API can control the default action that the code should take when a forwarding entry does not exist. This capability enables developers to program Network

Processors 343 without having to modify the loading function of the network processors code.

Network Processor Multicast Services APIs 321

provide a way to allocate multicast resources within the Network Processor 343. These multicast resources define how a packet is replicated from the Network Processor 343 into the Switch Fabric and from the Network Processor 343 to the attached media ports.

For portability among different operating systems, the Network Processor Services relies on a set of system services APIs (or OS) 322 that is mapped to the native API services of the targeted operating system 303.

System services APIs 322 operate in tandem with OS 303 and device driver 301 to provide dynamic portability of various external APIs 305, etc. to operate within the present hardware, firmware and software structure of network processor device driver 301. The present invention thus provides a way to easily port the Network Processor Services components by only modifying this one component. The system service API 322 allows OS 303 and processor independence by converting specific codes unique to that OS 303 and processor.

Extending across all three levels and interacting with each level and each functional block either directly or indirectly is the Operating System 303. Although the Network Processor Services APIs 305 are designed to be

hardware-architecture and operating-system independent, it is necessary for each functional block which requires OS control and/or interaction to be capable of communicating with OS 303.

5
The present invention provides the decomposition of network processor services for a general-purpose processor into individual functional component blocks, which enables the network processor services to be operating system and hardware (processor) independent. 10
The invention solves the problem of determining how to manage the network interfaces and how to decompose and fit the services into a system with existing and new interfaces by providing a unique device driver mechanism for managing a network processor. Use of functional component blocks makes it more flexible to add and remove new components by using generic services and layers along with an interpreter utility and a system services utility or OS. Those skilled in the art will appreciate the inherent benefits (or value) in decomposing network layers which were previously combined into component blocks and allowing for addition of additional functional utilities based on customer needs or network develop design. 20

25
It is also important to note that, although the present invention has been described in the context of a fully functional computer system, those skilled in the art will appreciate that the mechanisms of the present invention are capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular 30

type of signal-bearing media utilized to actually carry out the distribution. Examples of signal bearing media include, but are not limited to, recordable-type media such as floppy disks or CD ROMs and transmission-type media such as analogue or digital communications links.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

CLAIMS:

What is claimed is:

1 1. A method for providing scalable device driver
2 services within a control system of a network processor
3 services architecture comprising the steps of:

4 in response to determining a desired network
5 processor functionality, loading a plurality of
6 functional components that each provide a respective one
7 of a plurality of network processor services;

8 providing at least one utility interposed between
9 said plurality of functional components and an operating
10 system (OS) of said control system, that provide an OS
11 independent communication interface for said plurality of
12 functional components; and

13 in response to a receipt of a packet at said control
14 system, handling said packet utilizing one of said
15 plurality of individual software components by first
16 routing said packet through said utility, wherein said
17 packet is decoded into a common code understandable by
18 said OS and said one of said plurality of individual
19 software components.

1 2. The method of Claim 1, wherein said loading step
2 includes the step of loading external application
3 programming interfaces (APIs), low level APIs, and
4 physical transport interfaces of a device driver.

1 3. The method of Claim 2, further comprising the step
2 of loading a customer definable service component within
3 said external (APIs) that includes a customer desired
4 network service, which is operable within said network
5 processor services architecture.

1 4. The method of Claim 1, wherein said providing step
2 includes the step of providing a bi-directional
3 connection between said utility and said operating
4 system, one or more network processors, and each of said
5 functional components.

1 5. The method of claim 1, wherein said providing step
2 further comprises the step of linking a system services
3 utility to said operating system, wherein, said system
4 services utility operates to allow each of said
5 individual software coded components to communicate with
6 said OS.

1 6. The method of Claim 1, wherein said providing step
2 further provides a translation utility, which translates
3 all incoming and outgoing service requests into a common
4 network processor language to permit seamless connection
5 and correspondence between said one or more network
6 processors, said operating system, and each of said
7 functional components to enable handling of network
8 packets.

1 7. A computer program product for providing scalable
2 device driver services within a control system of a
3 network processor services architecture comprising:

4 a computer readable medium; and
5 program instructions on said computer readable
6 medium for:

7 implementing a plurality of individual software
8 components that each provide a respective one of a
9 plurality of network processor services;

10 providing at least one utility interposed between
11 said plurality of individual software components and an
12 operating system (OS) of said control system, that
13 provide an OS independent communication interface for
14 said plurality of individual software components; and

15 in response to a receipt of a packet at said control
16 system, handling said packet utilizing one of said
17 plurality of individual software components, by first
18 routing said packet through said utility, wherein said
19 packet is translated into a code understandable by said
20 OS and said one of said plurality of individual software
21 components.

1 8. The computer program product of Claim 7, wherein
2 program instructions for said loading step includes
3 program instructions for loading external application
4 programming interfaces (APIs), low level APIs, and
5 physical transport interfaces of a device driver.

1 9. The computer program product of Claim 8, further
2 comprising program instructions for loading a customer
3 definable service component within said external APIs
4 that includes a customer desired network service, which

5 is operable within said network processor services
6 architecture.

1 10. The computer program product of Claim 7, wherein
2 said program instructions for said providing step
3 includes instructions for providing a bi-directional
4 connection between said utility and said operating
5 system, one or more network processors, and each of said
6 functional components.

1 11. The computer program product of claim 7, wherein
2 said program instructions for said providing step further
3 comprises instructions for linking a system services
4 utility to said operating system, wherein, said system
5 services utility operates to allow each of said
6 individual software coded components to communicate with
7 said OS.

1 12. The computer program product of Claim 7, wherein
2 said program instructions for said providing step further
3 includes program instructions for a translation utility,
4 which translates all incoming and outgoing service
5 requests into a common network processor language to
6 permit seamless connection and correspondence between
7 said one or more network processors, said operating
8 system, and each of said functional components to enable
9 handling of network packets.

1 13. A control system of a network processor services
2 architecture comprising:

3 a plurality of individually loadable functional
4 components within a device driver of said control system
5 that each represents a network processor service;

6 at least one utility for enabling each of said
7 functional components to communicate with an operating
8 system (OS) of said control system, wherein said utility
9 responsive to a receipt of a request by said OS to
10 perform a network processor function, translates said
11 request into a call of a particular one of said plurality
12 of functional components that administers said network
13 processor services of one or more network processors; and

14 processing hardware that executes said OS, said
15 utility and said functional components.

1 14. The system of Claim 13, wherein said plurality of
2 functional components includes external application
3 programming interfaces (APIs), low level APIs, and
4 physical transport interfaces of a device driver.

1 15. The system of Claim 14, further comprising a
2 customer definable service component within said external
3 (APIs) that includes a customer desired network service,
4 which is operable within said network processor services
5 architecture.

1 16. The system of Claim 13, wherein said utility
2 includes a system services utility coupled to said
3 operating system that operates to allow each of said
4 individual software coded components to communicate with
5 said OS.

1 17. The system of Claim 13, wherein said utility
2 includes a translation utility, which translates all
3 incoming and outgoing service requests into a common
4 network processor language to permit seamless connection
5 and correspondence between said one or more network
6 processors, said operating system, and each of said
7 functional components to enable handling of network
8 packets.

1 18. A network processing services architecture
2 comprising:

3 one or more network processors; and
4 a control system, coupled to said one or more network
5 processors, having a scalable device driver for
6 controlling interactions with said one or more network
7 processors, wherein said scalable device driver is
8 comprised of a plurality of individual functional
9 software components including a plurality of functional
10 elements for network processing and a conversion utility
11 to enable compatibility between said plurality of
12 functional elements and said one or more network
13 processors to effectively communicate and interact within
14 any processor type and OS to effectuate the transfer of
15 frames and control information between said control
16 system and said one or more network processors.

1 19. The network processing services architecture of
2 Claim 18, further comprising a switch fabric coupled to
3 said one or more network processors.

1 20. A device driver program for a distributed
2 processing system comprising:

3 a plurality of system services programs providing
4 high level bi-directional communication between a common
5 code program and each of a network transport manager, a
6 network resource manager and a control message formatter,
7 said network transport manager providing low level
8 bi-directional communication to a physical transport
9 processor, said physical transport processor providing
10 primitive bi-directional communication to one or more
11 network processors.

ABSTRACT OF THE DISCLOSURE
A NETWORK PROCESSOR SERVICES ARCHITECTURE THAT IS
PLATFORM AND OPERATING SYSTEM INDEPENDENT

5 A system for providing a scalable processor and
operating system independent network processor services
architecture. The system includes a plurality of
portable and individualized functional components
representing particular segments of the control
processor's device driver. The functional components,
10 which include lower level and external APIs, carry out
the various network processor functions such as the
receipt and transfer of packets on the network, and other
functions required by the control processor to
communicate with and direct the network processor. The
15 functional components are designed to be adaptable to the
various types of processor architecture and operating
systems available and to permit customers or developers
to customize and expand the available network services.

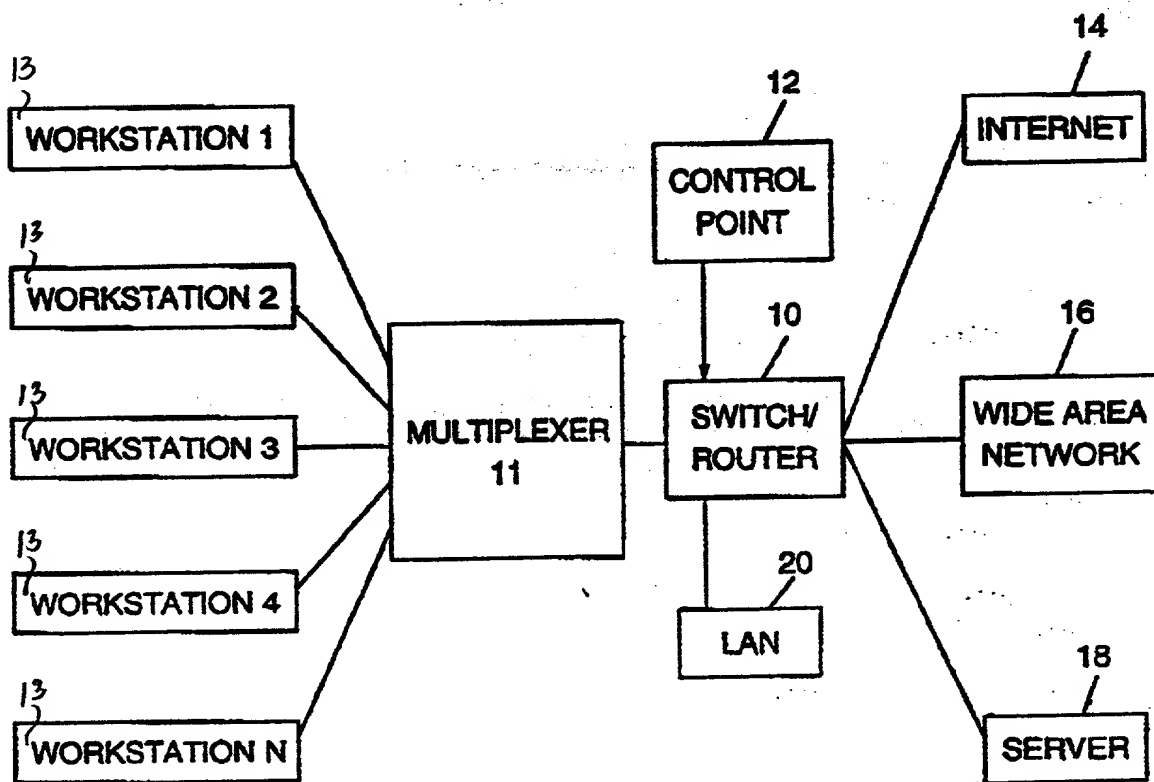


FIGURE 1A

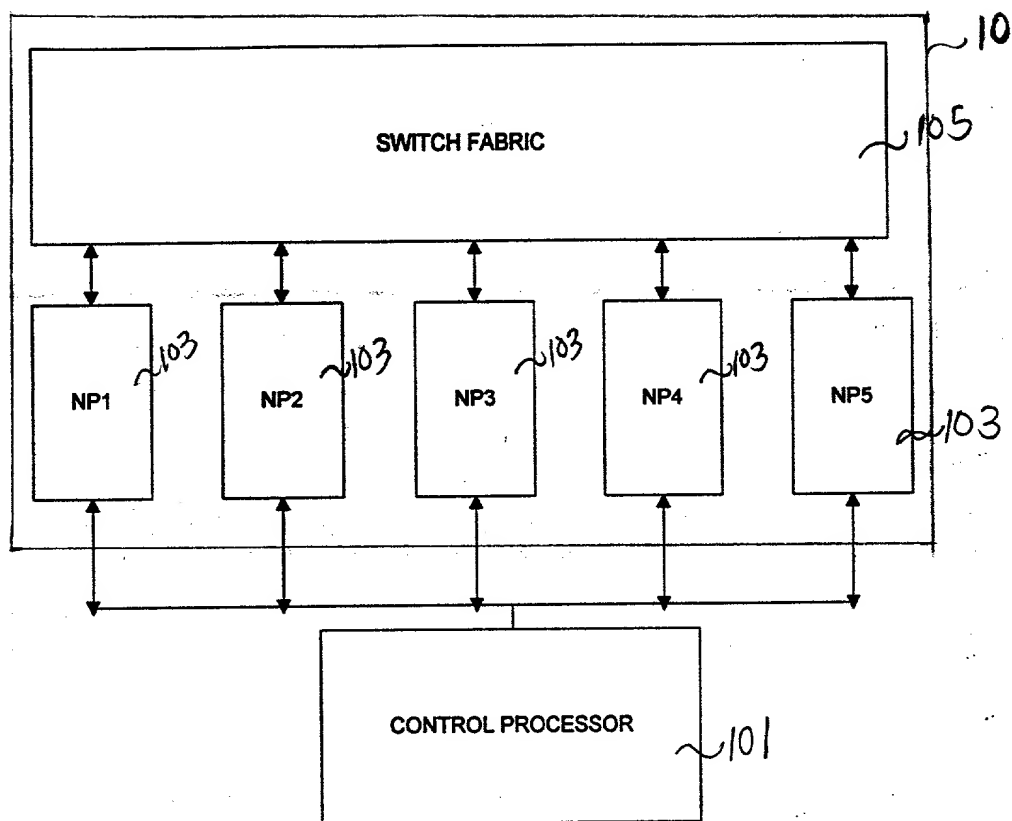


FIG 1B

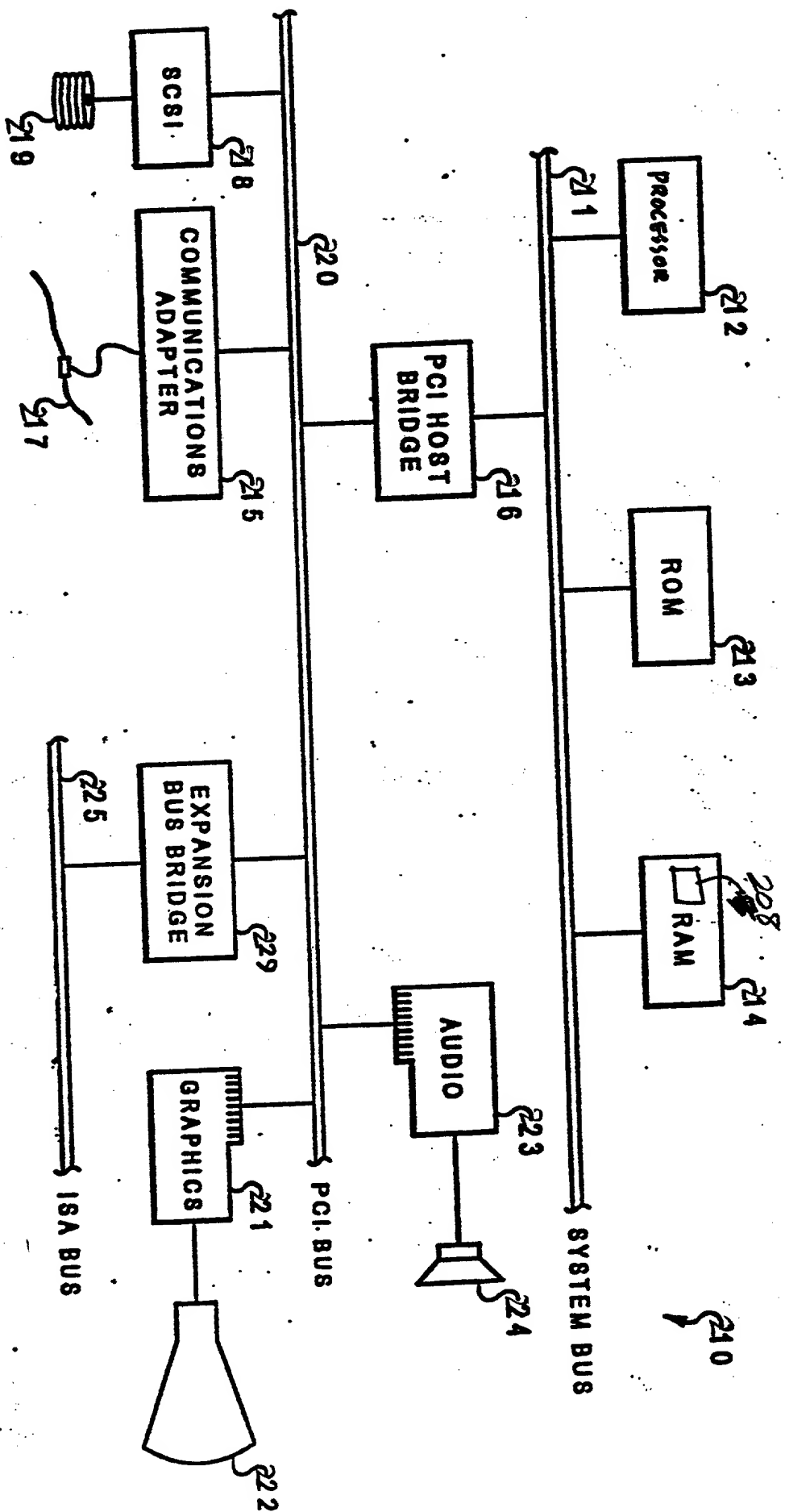
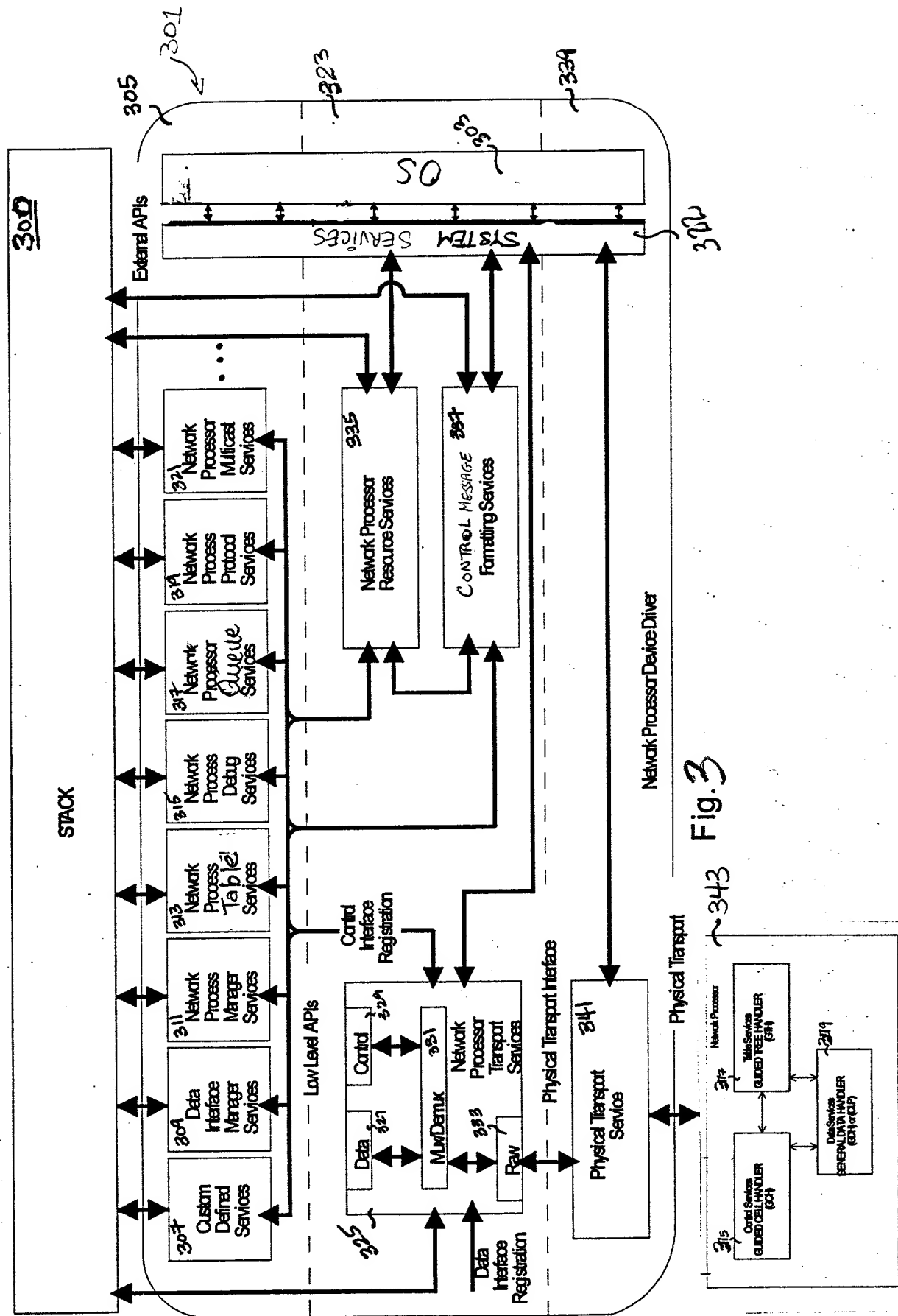


Fig. 2



DECLARATION AND POWER OF ATTORNEY FOR
PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

NETWORK PROCESSOR SERVICES ARCHITECTURE THAT IS PLATFORM AND OPERATING SYSTEM INDEPENDENT

the specification of which (check one)

X is attached hereto.

___ was filed on _____
as Application Serial No. _____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

Priority Claimed

(Number) (Country) (Day/Month/Year) ___ Yes ___ No

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal

Regulations, \$1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial #)

(Filing Date)

(Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

Joscelyn G. Cockburn, Reg. No. 27,069; Gerald R. Woods, Reg. No. 24,144; H. St. Julian, Reg. No. 30,329; John D. Flynn, Reg. No. 35,137; Kenneth Seaman, Reg. No. 28,113; Daniel E. McConnell, Reg. No. 20,360; Steve Tytran, Reg. No. P-45,846 (Agent); Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Andrew J. Dillon, Reg. No. Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Andrew J. Dillon, Reg. No. 29,634; Max Ciccarella, Reg. No. 39,454; Daniel E. Venglarik, Reg. No. 39,409; Jack V. Musgrove, Reg. No. 31,986; Brian F. Russell, Reg. No. 40,796; Steven Lin, Reg. No. 35,250; Matthew W. Baca, Reg. No. 42,277; Antony P. Ng, Reg. No. 43,427; John G. Graham, Reg. No. 19,563; Matthew S. Anderson, Reg. No. 39,093; Michael R. Barre, Reg. No. 44,023; Andrew Mitchell Harris, Reg. No. 42,638; Richard McCain, Reg. No. 43,785; Michael Noe, Reg. No. 44,975; and Sidney L. Weatherford, Reg. No. P-45,602.

Send correspondence to: Joscelyn (Josh) G. Cockburn, IBM CORPORATION, 3039 Cornwallis Road, Dept. 972/B656, Research Triangle Park, North Carolina 27709, and direct all telephone calls to Joscelyn (Josh) G. Cockburn, (919) 543-9036.

FULL NAME OF SOLE OR FIRST INVENTOR: Claude Basso

INVENTORS SIGNATURE: _____ DATE: _____

RESIDENCE: 7604 Percy Court
Raleigh, NC 27613

CITIZENSHIP: France

POST OFFICE ADDRESS: 7604 Percy Court
Raleigh, NC 27613

DOCKET NUMBER: RAL9-00-0029

FULL NAME OF SECOND INVENTOR: Philippe Damon

INVENTORS SIGNATURE: _____ DATE: _____

RESIDENCE: 1000 Smith Level Road, Apt. S8
Carrboro, NC 27510

CITIZENSHIP: France

POST OFFICE ADDRESS: 1000 Smith Level Road, Apt. S8
Carrboro, NC 27510

FULL NAME OF THIRD INVENTOR: Anthony Matteo Gallo

INVENTORS SIGNATURE: _____ DATE: _____

RESIDENCE: 3308 Corsham Drive
Apex, NC 27613

CITIZENSHIP: United States of America

POST OFFICE ADDRESS: 3308 Corsham Drive
Apex, NC 27613